POCKET COMPUTER NEWSLETTER

© Copyright 1983 SCELBI Publications

Issue 25 — June

NEW RADIO SHACK MODEL 100 SETS TREND

In case you have not yet seen one, the accompanying photograph illustrates what appears to be the hottest portable computer introduced to date: the new Radio Shack Model 100. Technical information on this unit was provided in Issue 24 of *PCN*. While this computer can not be classified as "pocket-sized", the compact portable fits easily in a briefcase or handbag. The unit is being warmly received by the buying public, apparently even beyond the expectation of Radio Shack's proud marketing department. It provides a lot of features that newcomers and old computer hands want, including a comfortable keyboard, large display and built-in modem complete with terminal (communications) software.

While Tandy Corporation normally does not release product-specific sales figures, *PCN* estimates that the firm has probably sold approximately 50,000 units in the first month following its introduction. Radio Shack may find itself hard-pressed to meet the demand for this model in the coming months as its popularity steadily mounts.

An indication of its extremely fast acceptance by the public is the fact that the Compuserve computer network has already created

(continued on page 4)

Another *Personal Information*™ product from SCELBI Publications.

PC-1500/PC-2 CASSETTE STORAGE

The cassette storage capabilities of the Sharp PC-1500 and Radio Shack PC-2 pocket computers are only partially detailed in the manuals supplied with those units. Furthermore, the manuals are confusing in some respects when describing these operations. This article is intended as a summary and clarification of the cassette operations. Thanks for this article go to: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.*

BASIC Programs

The CSAVE statement, which is well covered in the manuals, may be used to save BASIC programs manually (in the RUN or PRO mode) or as a statement within a program. Its execution records the information beginning at the address pointed to by the START OF BASIC pointer and ending at the address pointed to by the END OF BASIC pointer. (These pointers are located at &7865-66 and &7867-68.) When the saved program is reloaded into the computer using CLOAD or CHAIN, these pointers are automatically reset to the values they had when the program was CSAVEd.

The CLOAD statement may be executed manually in either RUN or PRO mode. CHAIN is similar to CLOAD in effect, except that it is usable only within a program and it automatically begins execution of the program loaded from cassette. (The manuals cover CHAIN well.)

RESERVE Memory

If CSAVE is executed when the computer is set to REServe mode, the contents of REServe memory are recorded. A CLOAD statement, executed in REServe mode, loads the stored codes back into REServe memory.

Machine Language Codes

Machine language programs may be saved with the CSAVE M statement. This is not discussed in the instruction manuals. This command takes the form: CSAVE M "FILE NAME"; expr 1, expr 2, expr 3. The first and second expressions are interpreted as the beginning and ending addresses of the code that is to be recorded. The third expression, which is optional, is a memory address at which the computer will automatically begin execution of machine language codes. This will occur as soon as the recorded material has been loaded into memory using an appropriate CLOAD M directive.

This auto-execution feature may be used, for example, to save both the machine language and BASIC portions of a "hybrid" program so as to permit loading of both portions with a single operation. This is accomplished by the inclusion of a short machine language program that sets the BASIC pointers as soon as loading from the cassette has been completed.

An example should help clarify this operation. Suppose your program includes a machine language portion in the area &38C5 to &38FF and the START OF BASIC pointer is set to &3900. Assume also that your END OF BASIC pointer is set to &3B85 (as determined by peeking at &7867-68). In this situation we will need a machine language routine that will be automatically executed after a CLOAD M directive has been completed. This machine language routine must set the START OF BASIC, END OF BASIC and START OF EDIT pointers. (The START OF EDIT pointer is discussed later in this article.)

An ideal location for this machine language routine is in the area immediately following the BASIC program. Thus, for this example, it would begin at &3B86. The machine language routine will end with a call to the base-page-addressed routine 42. This routine sets the PC to the "READY" state. Such a routine, beginning at &3B86 might appear as shown in the accompanying listing. It would be loaded into memory using POKE directives. (Remember that the example listing is just that! The actual pointer settings you would use would depend on the size and location of your BASIC program, etc.)

Example *Routine Sets BASIC Pointers Then Exits to PC "Ready" State.*

Opcodes	Mnemonics	Comments
B5 39	LDA #39	Set START OF BASIC
AE 78 65	STA 7865	pointer
AE 78 69	STA 7869	as well as the
B5 00	LDA #00	START OF EDIT
AE 78 66	STA 7866	pointer to the
AE 78 6A	STA 786A	value &3900
B5 3B	LDA #3B	Set END OF BASIC
AE 78 67	STA 7867	pointer to the
B5 85	LDA #85	value &3B85
AE 78 68	STA 7868	
CE 42	CALL #42	Set computer to "Ready"

Continuing to discuss the example, the cassette recording in this case would be made by execution of the directive:

CSAVE M "FILE NAME"; &38C5, &3BA1, &3B86

Thus, all information for both the machine language and BASIC portions of the program, along with the pointer-setting routine, would be recorded.

Loading this recording at a later time using the CLOAD M command will result in the BASIC pointers being set automatically as soon as the loading is completed.

If you wanted to save the contents of REServe memory also, then the only change needed in the preceding example would be to have the first expression in the CSAVE M directive specify &3800 instead of &38C5. (It is not necessary to set pointers for REServe memory in such a situation.)

Merged Programs

The instruction manuals can easily lead one to believe that the use of MERGE is a lot more trouble than it is worth. Such is really not the case.

Suppose you have several programs already saved on cassette tape and you would like to be able to use all of them by having them available in memory at one time. The MERGE directive permits placing any number of previously recorded programs into the computer at once (assuming, of course, that you do not run out of memory). No problems are caused by the overlapping or repetition of line numbers in the various programs as each merged program operates independently. The programs that are MERGEd together are separated from each other by a "stop" byte (code &FF) that effectively blocks searches for line numbers beyond the end of each program section. These stop bytes, however, *do not block searches for labels!* This feature of PC-1500/PC-2 BASIC is what makes it possible to execute a number of different programs that are stored in memory at one time.

It is important to note, however, that when two or more programs are in memory at one time, that the following two rules must be observed:

1. Execution of any program (other than the first one) *must* be begun with a *label* address. Execution of the program or routine may then be initiated by manually telling the computer to RUN "LABEL" or GOTO "LABEL" or, when appropriate, by using the DEFine key. (Of course, care must be taken to ensure that each label only appears in the combined programs one time. If there are multiple occurrences then the first appearance of the specified label will be utilized.)

2. The only program that may be edited is the last one loaded using the MERGE directive. Thus, any programs that do not begin with a label should be modified to begin with one. This can be done immediately after the program has been loaded, but *before* merging another section.

The pointer located at address &7869-6A has been identified previously in *PCN* as the BEGINNING OF MERGED PROGRAM or START OF PROGRAM marker. I hereby propose calling it START OF EDIT since its function is to locate the place at which editing of program lines may begin. When merging is not involved, its contents are

identical to the START OF BASIC pointer. After a MERGE it is automatically set to point to the beginning of the program that has just been loaded by the MERGE directive.

Knowing this information it becomes possible to alter memory so that two merged programs can be combined into a single "editable" program. Note that this is only possible if the lines in the combined programs are numbered consecutively. Here is how it could be done:

1. CLOAD the first program (if not already in memory).
2. Determine (using PEEK) the address contained in the END OF BASIC pointer (&7867-68). Reduce this value by one and POKE it into &7867-68.
3. Load the second program using MERGE.
4. Execute POKE &7869, PEEK &7865, PEEK &7866.

Recording of Data

There are six types of variables that can be recorded with PRINT # statements:

1. A single undimensioned numeric variable.
2. A single undimensioned string variable.
3. The complete set of "fixed" string variables (A through Z), specified as @(*).
4. The complete set of "fixed" string variables (A$ through Z$), specified as @$(*).
5. A complete numeric array (excluding "fixed" variables), specified by using * as a subscript.
6. A complete string array (excluding "fixed" string variables), specified by using * as a subscript.

During execution of a PRINT # statement, the type of variable in each case is recorded along with the data. (The name of the variable being recorded is not saved on tape.)

When INPUT # is used to load data from tape, each specification must match the type of the corresponding recorded data. Thus a recording made using the statement:

```
PRINT #"FILE NAME"; AC, D(*), @$(*), J$
```

could be loaded with:

```
INPUT #"FILE NAME"; K, A(*), @$(*), D2$
```

provided that A() was dimensioned with the same parameters as D() and that D2$ was properly defined.

Whenever variables in "main" memory (that is, other than variables A through Z and A$ through Z$) are named in an INPUT # statement, those variables must have been defined prior to execution of INPUT #. Furthermore, if dimensioned, they must have the same dimensions as those previously recorded. If they are dimensioned string variables, they must also agree in length. Any mismatch will produce an ERROR 43 message.

It is of interest to note that CSAVE M may be used to record variables. Of course, if this is done then CLOAD M must be used to recover those variables from tape. As an illustration, you could use:

```
CSAVE M "FILE NAME"; &7900, &79CF
```

to save variables Z through Z. A CLOAD M command would then load them back into the computer. The advantage of using this method is that it is faster in operation than using PRINT # and INPUT #.

Assorted Information

As a summary, the four kinds of recordings possible are as follows:

1. REServe memory, saved with CSAVE, loadable only by CLOAD.
2. BASIC program, saved with CSAVE, loadable using CLOAD or MERGE or CHAIN.
3. Coded data (machine language routines, etc.) saved by CLOAD M, loadable using CLOAD M.
4. Variables, saved using PRINT #, loadable using INPUT #.

Remember that the PC distinguishes amongst all four of the above when reading from a tape, so that the same file name may be used for each different type of recording.

CLOAD? may be used to verify recordings made by CSAVE (either REServe memory or BASIC programs), but it cannot be used to verify recordings made using CSAVE M or PRINT #.

Both CSAVE M and CLOAD M may be included as statements within a program.

A second cassette recorder may be addressed using CSAVE-1, CSAVE M-1, PRINT #-1, CLOAD-1, CLOAD?-1, MERGE-1, CLOAD M-1, INPUT #-1, or CHAIN-1 statements.

The inclusion of a file name in cassette-related statements is optional except in the two following instances:

1. PRINT # and INPUT # statements in which the first variable named is a string variable.
2. CHAIN statements in which a line number or label is specified.

When a file name is left out or "" is used as a file name, the CLOAD, CLOAD M, CLOAD?, MERGE, CHAIN and INPUT # statements cause initiation of a search for a recorded file of the appropriate type. The first such file found is loaded.

If a CHAIN statement specifies a line number then it must end the program line in which it appears.

NORMAL PROBABILITY USING MACHINE LANGUAGE

Norlin Rober submitted this program that illustrates an example of the use of machine language to calculate a fairly complicated mathematical function. If you need to perform such calculations frequently, the time saved by using a machine language routine can be quite significant.

A number of short algorithms are commonly used for calculating Normal (Gaussian) probability. The most commonly used procedures are those developed by Hastings. For example, Sharp's POCKET COMPUTER APPLICATIONS MANUAL for the PC-1211 uses the "best" of these. It correctly provides the first seven places following the decimal point. Note though, that for values of Z greater than 5.2,

Program *Normal Probability in PC-1500/PC-2 Machine Language.*

```
38C5 48 79 4A C8  3949 41 DD 8E 0F
38C9 BE DC 20 BE  394D EB 7A 12 36
38CD F5 97 CD 80  3951 CD 7E CD 6A
38D1 BE F0 19 FD  3955 4A 12 B5 44
38D5 58 B5 4B FD  3959 41 B5 66 0E
38D9 CA BE F7 17  395D F0 CD 68 F0
38DD FD 88 CD 7E  3961 E6 F4 7B 60
38E1 BE F1 D4 FD  3965 CD 10 00 CD
38E5 0A BE F7 17  3969 58 A5 7B 61
38E9 CD 7E 58 79  396D DF AE 7B 61
38ED 5A 70 BE F7  3971 99 15 CD 68
38F1 11 BE DB F5  3975 F0 CD 6E E6
38F5 CD 70 EB 7A  3979 CD 30 CD 7E
38F9 10 FF EB 7A  397D A5 79 C9 8B
38FD 11 80 68 00  3981 07 CD 6A EF
3901 6A 15 F6 7B  3985 7A 01 80 F0
3905 60 BE F7 37  3989 5A 78 58 79
3909 A5 7A 00 B7  398D BA F7 11 BE
390D 80 83 80 B7  3991 F0 19 CD 80
3911 01 83 39 A5  3995 8E 05 CD 68
3915 7A 02 B7 16  3999 CD 7E E6 F4
3919 83 16 EF 7B  399D 7B 60 CD 10
391D 61 0A 8E 6F  39A1 00 CD 66 CD
3921 FF 80 21 71  39A5 58 CD 6A F0
3925 47 24 09 52  39A9 A5 7B 61 DF
3929 FF 00 39 89  39AD DF AE 7B 61
392D 42 28 04 01  39B1 DF 99 1D E6
3931 B7 25 83 18  39B5 48 79 4A C8
3935 EF 7B 61 10  39B9 BE DC 20 CD
3939 EB 7A 12 42  39BD 7E E6 CD 30
393D EB 7A 13 50  39C1 CD 7E CD 62
3941 CD 7E CD 6A  39C5 EF 7A 12 F0
3945 4A 12 B5 60  39C9 9E 47
```

where probabilities are less than 1E-7, the calculated results are meaningless.

The routine used in the accompanying machine language program gives a full ten digits of accuracy, regardless of the value of Z. (When Z exceeds 21.165, the probability is less than 1E-99, so the result underflows to zero.) Probabilities are calculated with a Taylor series for absolute values of Z less than 1.6, and with a continued fraction for larger values of Z. Execution is fast, taking less than a second.

You can enter the machine language program using a Monitor program or by using BASIC POKEs. (The program is *relocatable*, so if you wish you may start it at some address other than &38C5 without making any modifications. Just remember to call the right address when accessing the routine if you do move it elsewhere.)

To use the program, just follow these steps:

1. The value of Z to be used must be assigned to the variable Z.
2. Execute CALL &38C5 (or the starting address if you relocate the machine language routine).
3. Display the variable P to obtain the probability that a normally distributed random variable exceeds Z.

As a bonus, the variable O contains the ordinate of the normal curve for the given Z!

You can test the program's operation (to verify proper loading of the machine language codes) with the following examples:

Value of Z	Resulting P	Resulting O
-2	9.772498681E-01	5.399096651E-02
1.3	9.680048459E-02	0.171368592
4	3.167124183E-05	1.338302256E-04

ROUTINE DELETES LINES FROM BASIC PROGRAMS

John Norton, % Pencept, Inc., 39 Green Street, Waltham, MA 02154, submitted this program that makes it easy to remove groups of lines from a BASIC program. Here is what John has to say about his routine:

I have often wished for a simple way to eliminate sections of a BASIC program in a Sharp PC-1500/Radio Shack PC-2. Trying to remove a number of lines by specifying each line number is just too slow, cumbersome and subject to error. By capitalizing on the material that has appeared in *PCN*, I have been able to create a routine that will permit the deletion of a group of lines from a BASIC program.

To use this routine, it must be loaded into memory prior to your development of a BASIC program. Your BASIC program is then created (starting with a line number greater than 14 as this is the highest line number in the deletion routine). The routine uses a JUMP statement so that a RUN directive causes the PC to skip over it and execute the user's program.

The deletion routine may be utilized at any time by pressing the DEFine and D keys. It begins by prompting for two items: the line number at which to begin deleting and the last line number to be deleted. If, for example, you had a program starting at line 500, with a last line number of 850, and you wanted to delete all the lines from number 722 through 751, you would proceed as follows:

What you type	What the display shows
DEF/D (ENTER)	DELETE FROM:
722 (ENTER)	DELETE TO:
751 (ENTER)	WORKING ...

That is all there is to it! After these inputs, the routine will issue a beep each time it encounters a line less than 722. It will signal when it has found line 722, when it finds line 751, and when it does the deletion of the material between those lines.

The routine operates as follows: It marks the memory location at which line 722 starts. It then re-writes every byte from the first line after 751 to the end of the program, into memory starting from the marked byte (where line 722 started).

It is important to note the following: The routine will not attempt to delete itself. It does this be starting to look for lines in memory after itself. It does this by "knowing" exactly how long it is. Thus, the routine should not be altered by as much as a single character! Make sure you load it exactly as shown in the accompanying listing!

The routine uses default values as follows: If no value is given in response to the prompt "DELETE TO", then the "DELETE FROM" value is used. Thus, to delete a single line from memory, just type: DEF/D ### (ENTER) (ENTER), where ### represents the line number to be deleted.

In the event that the starting line is specified as larger than the ending line, the starting line is not found or the ending line is not located, the routine will display an error message such as "START NOT FOUND" or "END NOT FOUND" and cease operating.

To delete the routine itself from memory, type an asterisk (*) immediately before entering DEFine/D. This will cause the computer to restructure pointers so that the deletion routine is effectively erased!

Program *Line Deletion*

```
1:GOTO 16
2:"D":AREAD DP$:
  IF DP$="*"LET
  D1=59:D2=149:
  GOTO 14
3:CLEAR :WAIT 0:
  INPUT "DELETE
  FROM:";DF:DT=D
  F:INPUT "DELET
  E TO:";DT:
  PRINT "WORKING
  ..."
4:DM=256*PEEK &7
  865+PEEK &7866
  +720:DB=DM:IF
  PEEK DM=255LET
  DM=DM+1:DB=DM
5:DN=256*PEEK DM
  +PEEK (DM+1):
  IF PEEK DM=255
  OR DN>DFBEEP 5
  ,60:WAIT :
  PRINT "NO STAR
  T":END
6:BEEP 1,40:IF D
  N<DFLET DM=DM+
  PEEK (DM+2)+3:
  DB=DM:GOTO 5
7:BEEP 5,30:
  PRINT "START F
  OUND"
8:IF DN=DTLET DM
  =DM+PEEK (DM+2
  )+3:GOTO 11
9:IF PEEK DM=255
  OR DN>DTBEEP 5
  ,60:WAIT :
  PRINT "NO END"
  :END
10:BEEP 1,20:DM=D
  M+PEEK (DM+2)+
  3:DN=256*PEEK
  DM+PEEK (DM+1)
  :GOTO 8
11:BEEP 5,10:
  PRINT "...DELE
  TING":DE=256*
  PEEK &7867+
  PEEK &7868:FOR
  DI=0TO DE-DM:
  BEEP 1,0
12:POKE DB+DI,
  PEEK (DM+DI):
  NEXT DI:POKE &
  7867,INT ((DB+
  DI)/256),((DB+
  DI)AND &FF)
13:FOR DI=20TO 0
  STEP -1:BEEP 1
  ,DI:NEXT DI:
  END
14:D2=D2+((PEEK &
  7865<>PEEK &78
  69)OR (PEEK &7
  866<>PEEK &786
  A))
15:POKE &7865,D1,
  D2:POKE &7869,
  D1,D2:END
16:REM *MAIN PROG
  RAM*
STATUS 1          721
```

(Model 100 . . . continuation from page 1)

a "SIG" (Special Interest Group) devoted exclusively to Model 100 enthusiasts. In the first two weeks following its activation, some 1600 users have joined this SIG in order to use the Model 100-related information base. The Compuserve Model 100 SIG (which can be accessed by Compuserve users typing GO PCS-154 at the command (!) prompt) has already had some 30 programs contributed to its user library. These range from several simple "demo" programs on up to useful utilities such as memory dumps, disassemblers, printer formatting

routines (that will format the output from text files). There are also programs that generate graphics and produce music.

Another indicator of the success of this unit is that already three (that's right: 3!) organizations have announced that they will be publishing magazines that will cater to Model 100 afficianados. The three firms that have announced such publications to date are:

PORTABLE 100 MAGAZINE
21 Elm Street
Box 597
Camden, ME 04843

PORTABLE COMPUTING MAGAZINE
9529 U.S. Highway 42
P.O. Box 209
Prospect, KY 40059

BRIEFCASE PORTABLE
560 South Hartz Avenue - Suite 447
Danville, CA 94526

If you are interested in any of these publications, it is suggested that you contact them right away. Some of them are offering special "pre-publication" subscription rates that may be withdrawn soon.

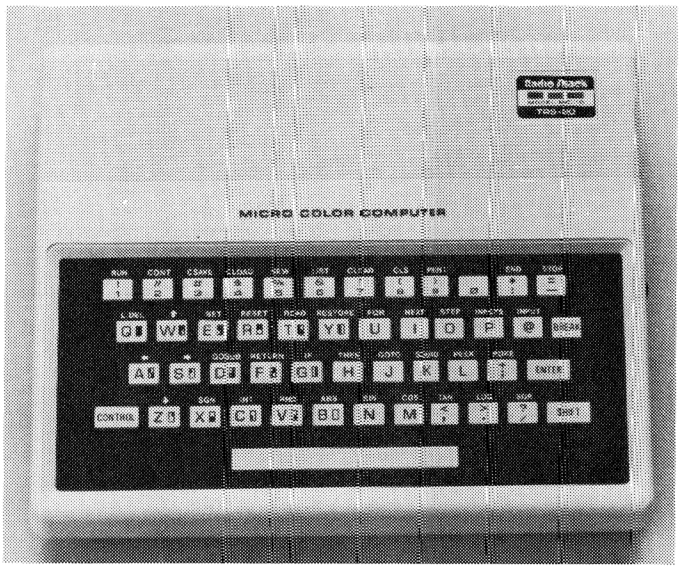
RADIO SHACK ANNOUNCES MICRO COLOR COMPUTER

While it might not technically be termed "portable" because it requires the use of 110 volt a.c. for power, the new MC-10 from Radio Shack is pretty small. Measuring just 7 by 8 by 2 inches and weighing 29.5 ounces (a shade under two pounds), the unit is easily transported from one location to another.

While equipped with a full keyboard and an RF moudulator, the user must provide a television set as a display. The MC-10 can generate eight different colors for graphics and display text in a 32-character, 16-line format. The unit is programmable in BASIC and comes equipped with 4K of memory, expandable to 20K. It also is equipped with a serial I/O port to provide connection to modems or printers as well as a cassette I/O port for loading and saving programs on cassette tape.

An easy-to-read operator's manual is provided with the MC-10. It includes instruction in the use of Microsoft color BASIC especially aimed at beginners as well as a quick-reference guide for more experienced programmers. It is claimed that almost all standard color BASIC programs that work on a 4K Radio Shack TRS-80 Color Computer can be keyed in and run on the new MC-10 with minor changes.

The MC-10 is low-priced: just $119.95 at U.S. Radio Shack Computer Centers and other Radio Shack outlets!



GRAPH PROGRAM FOR THE PC-1500/PC-2

This program, provided by *John Norton*, enables the Sharp PC-1500 or Radio Shack PC-2 equipped with a printer/plotter to draw a variety of graphs. Straight lines such as that expressed by the equation Y=3*X+7, curves such as Y=SIN X, functions with extremes such as Y=1/X, functions with asymptotes and even pseudo-functions such as Y=RND 100 can all be drawn through the use of this program. Want to plot using polar coordinates? It can do that, too! The program allows you to define the region of the coordinate plane to be drawn, to select rectangular or polar coordinates, the increment (accuracy) size, and X and Y scaling factors.

The program that accomplishes all this is shown in the accompanying listing. To utilize the program you must substitute the function that you wish to graph in line 50. For example, if you wanted to examine the function Y=SIN X, you would have line 50 read:

```
50 Y=SIN(X)
```

Notice that this example assumes a rectangular coordinate system. That is, the distance vertically is a function of the horizontal distance. If you were interested in a polar coordinate graph of the SIN function, you would have line 50 appear as:

```
50 R=SIN(T)
```

In this case, R would represent the radius and T would indicate the angle (theta). When using polar (as contrasted to rectangular) coordinates, the distance of the plotted point from the origin is a function of the angle. Therefore, for polar plotting express equations using R = (a function of T). When using rectangular coordinates, express Y = (a function of X).

Once you have placed the desired function in line 50, the program is ready to be used. Just issue the RUN command. Respond to the prompts for rectangular or polar coordinates (R or P), minimum and maximum values of X and Y (or theta) and the X (or theta) increment size. You will also be prompted for X and Y scaling factors.

Once the prompts have been responded to, the program will display pertinent information about the graph it is to draw. It then draws a window of the graph, X and/or Y axes (if they occur within the window), places scaling marks as appropriate and then graphs the function.

If the plot does not appear as you expected it to, you might want to check the accompanying table that lists typical graphing problems. A second table indicates default values that will be used if you respond to a prompt by just pressing the ENTER key.

A selection of drawings produced by the program and the functions and parameters used to generate these drawings is also provided for checking purposes (and as a general illustration of the program's capabilities). Try experimenting. You can get a lot of interesting results!

Table *Typical Graphing Problems*

PROGRAM OUTPUT	POSSIBLE PROBLEM
1. Nothing plotted in window.	1a. Window too small or large or is not over the proper portion of the graph.
	1b. RAD/DEG/GRAD not set properly.
	1c. Increment value too large.
2. Lines appearing where they should not exist.	2. Increment value too large.
3. Curves appear squared.	3. Increment value too large.
4. Program seems to take too long to plot.	4a. Increment value too small.
	4b. Function is producing an error causing program to continuously loop.

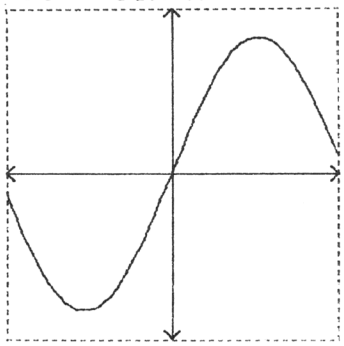
Program *Graph*

```
 10:REM  ** GRAPH
    -- Written by
    John Norton **
 20:"A":WAIT 0:F1=
    0:F2=0:RF=1000
    000:E=2.718281
    828:GOTO 100
 30:BEEP 1,1
 40:ON ERROR GOTO
    850
 50:REM -FUNCTION-
 60:IF G$="P"LET X
    =R*COS T
 70:IF G$="P"LET Y
    =R*SIN T
 80:PRINT "("+STR$
    (INT (RF*X+.5)
    /RF)+","+STR$
    (INT (RF*Y+.5)
    /RF)+")"
 90:RETURN
100:G$="R":INPUT "
    Type of graph
    (R or P): ";G$
110:X1=-10:INPUT "
    X-minimum:";X1
120:X2=-X1:INPUT "
    X-maximum:";X2
130:I=(X2-X1)/75:
    IF G$="R"INPUT
    "X-increment:"
    ;I
140:Y1=X1:INPUT "Y
    -minimum:";Y1
150:Y2=-Y1:INPUT "
    Y-maximum:";Y2
160:T1=0:IF G$="P"
    INPUT "Theta m
    inimum:";T1
170:T2=2*π:IF G$="
    P"INPUT "Theta
     maximum:";T2
180:IF G$="P"LET I
    =(T2-T1)/75:
    INPUT "Theta-i
    ncrement:";I
190:S1=0:INPUT "X-
    scaling:";S1
200:S2=S1:INPUT "Y
    -scaling:";S2
210:CLS :TEXT :
    COLOR 3:LLIST
    50:LF 10:COLOR
    0
220:CSIZE 1
230:IF G$="R"
    LPRINT "RECTAN
    GULAR COORDINA
    TES"
240:IF G$="P"
    LPRINT "POLAR
    COORDINATES"
250:LF 1:CSIZE 2
260:LPRINT "X: ";X
    1;",";X2
270:LPRINT "Y: ";Y
    1;",";Y2
280:IF G$="P"
    LPRINT "O";LF
    -1:LPRINT "-:
    ";T1;",";T2
290:LPRINT "INC: "
    +STR$ (INT (RF
    *I+.5)/RF)
300:IF S1>0LPRINT
    "X-S: ";S1
310:IF S2>0LPRINT
    "Y-S: ";S2
320:LF -6-(G$="P")
    -(S1>0)-(S2>0)
330:GRAPH :SORGN :
    COLOR 1
340:LINE (0,0)-(21
    6,216),2,1,B
350:PY=X1/(X1-X2)*
    216
360:IF (PY<0)OR (P
    Y>216)GOTO 440
370:F1=1
380:LINE -(PY,216)
    ,9
390:RLINE -(-5,-5)
    ,0:RLINE -(5,5
    ),0
400:RLINE -(5,-5),
    0:RLINE -(-5,5
    ),0
410:LINE -(PY,0)
420:RLINE -(-5,5),
    0:RLINE -(5,-5
    ),0
430:RLINE -(5,5),0
    :RLINE -(-5,-5
    ),0
440:PX=Y1/(Y1-Y2)*
    216
450:IF (PX<0)OR (P
    X>216)GOTO 530
460:F2=1
470:LINE -(0,PX),9
480:RLINE -(5,5),0
    :RLINE -(-5,-5
    ),0
490:RLINE -(5,-5),
    0:RLINE -(-5,5
    ),0
500:LINE -(216,PX)
510:RLINE -(-5,5),
    0:RLINE -(5,-5
    ),0
520:RLINE -(-5,-5)
    ,0:RLINE -(5,5
    ),0
530:IF (S1+S2=0)OR
    (F1+F2=0)GOTO
    700
540:COLOR 3
550:IF (S2*F2)=0
    GOTO 620
560:K=0
570:K=K+S2
580:IF K<=Y2LINE (
    PY-2,(K-Y1)/(Y
    2-Y1)*216)-(PY
    +4,(K-Y1)/(Y2-
    Y1)*216),0:
    GOTO 570
590:K=0
600:K=K-S2
610:IF K>=Y1LINE (
    PY-2,(K-Y1)/(Y
    2-Y1)*216)-(PY
    +4,(K-Y1)/(Y2-
    Y1)*216),0:
    GOTO 600
620:IF (S1*F1)=0
    GOTO 690
630:K=0
640:K=K+S1
650:IF K<=X2LINE (
    (K-X1)/(X2-X1)
    *216,PX+3)-((K
    -X1)/(X2-X1)*2
    16,PX-3),0:
    GOTO 640
660:K=0
670:K=K-S1
680:IF K>=X1LINE (
    (K-X1)/(X2-X1)
    *216,PX+3)-((K
    -X1)/(X2-X1)*2
    16,PX-3),0:
    GOTO 670
690:COLOR 2
700:CLS :RANDOM :X
    =X1:T=T1:GOSUB
    30
710:G2=(Y-Y1)/(Y2-
    Y1)*216
720:IF G$="P"LET G
    1=(X-X1)/(X2-X
    1)*216
730:IF (G1<0)OR (G
    1>216)LET F=1:
    GOTO 760
740:IF (G2<0)OR (G
    2>216)LET F=1:
    GOTO 760
750:LINE -(G1,G2),
    9
760:COLOR 2:X=X+I:
    T=T+I
770:IF (X>X2)AND (
    G$="R")GOTO 86
    0
780:IF (T>T2)AND (
    G$="P")GOTO 86
    0
790:GOSUB 30
800:P2=(Y-Y1)/(Y2-
    Y1)*216:P1=(X-
    X1)/(X2-X1)*21
    6
810:IF (P1<0)OR (P
    1>216)OR (P2<0
    )OR (P2>216)
    LET F=1:GOTO 8
    40
820:IF F=1LET F=0:
    GOTO 840
830:LINE (G1,G2)-(
    P1,P2),0
840:G1=P1:G2=P2:
    GOTO 760
850:F=1:BEEP 1,20:
    GOTO 760
860:LINE -(0,0),9
870:TEXT :LF 10+(S
    1>0)+(S2>0)
880:END
STATUS 1       2263
```

Table *Default Values for Graph Program*

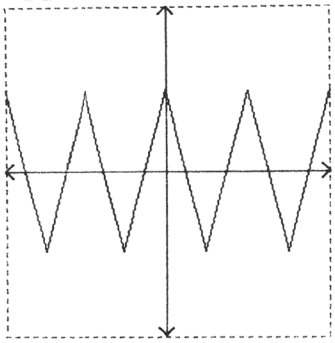
INPUT REQUESTED	DEFAULT VALUE
Rectangular/Polar Coord's	Rectangular (R)
X - minimum	-10
X - maximum	- X - minimum
X - increment	(X max - X min)/75
Y - minimum	Y minimum
Y - maximum	- Y - minimum
Theta min/max	0, 2π
Theta increment	(T max - T min)/75
X / Y scaling	0, X

50:Y=SIN X



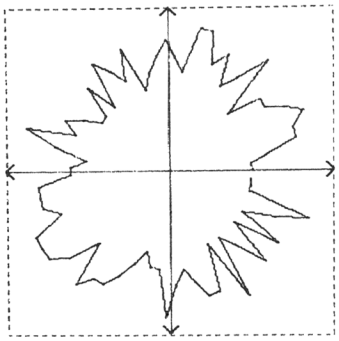
RECTANGULAR COORDINATES
X: -3, 3
Y: -1.2, 1.2
INC: 0.1

50:Y=COS X



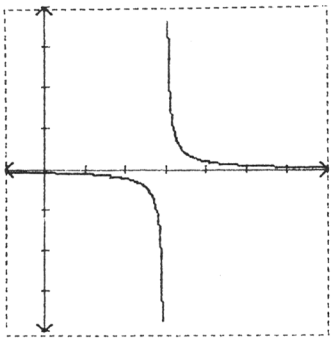
RECTANGULAR COORDINATES
X: -12.56, 12.56
Y: -2, 2
INC: 1.57

50:R=10-RND 5



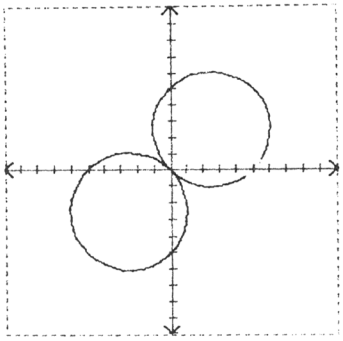
POLAR COORDINATES
X: -10, 10
Y: -10, 10
θ: 0, 6.283185307
INC: 0.083776

50:Y=1/(X-3)



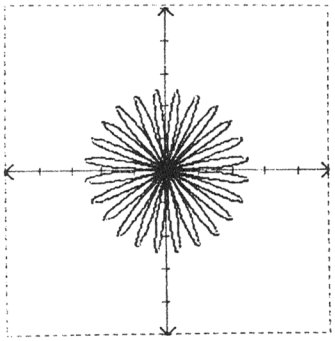
RECTANGULAR COORDINATES
X: -1, 7
Y: -20, 20
INC: 0.021333
X-S: 1
Y-S: 5

50:R=ABS (SIN T+ COS T)



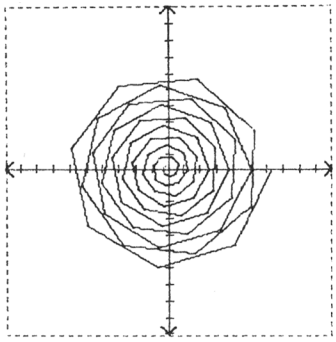
POLAR COORDINATES
X: -2, 2
Y: -2, 2
θ: 0, 6.283185307
INC: 0.083776
X-S: 0.2
Y-S: 0.2

50:R=SIN (6*T)* COS (6*T)



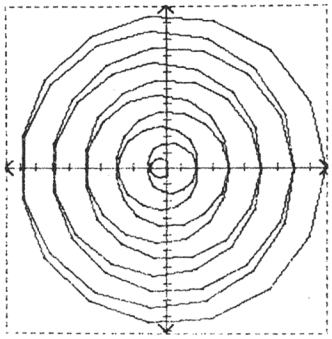
POLAR COORDINATES
X: -1, 1
Y: -1, 1
θ: 0, 6.283185307
INC: 0.013963
X-S: 0.2
Y-S: 0.2

50:R=SIN T+T



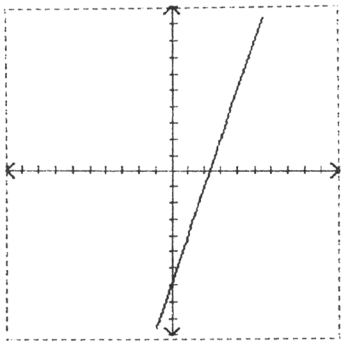
POLAR COORDINATES
X: -100, 100
Y: -100, 100
θ: 0, 62.83185307
INC: 0.837758
X-S: 10
Y-S: 10

50:R=ABS (10*Π-T)



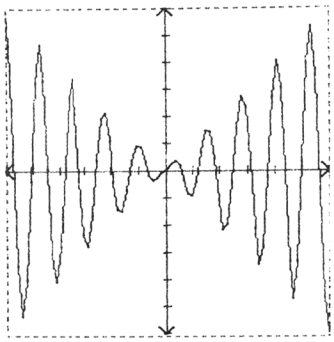
POLAR COORDINATES
X: -31.416, 31.416
Y: -31.416, 31.416
θ: 0, 62.84
INC: 0.418879
X-S: 3.141592654
Y-S: 3.141592654

50:Y=3*X-7



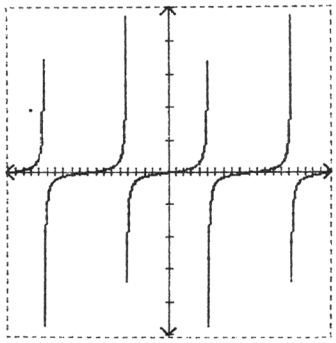
RECTANGULAR COORDINATES
X: -10, 10
Y: -10, 10
INC: 0.266667
X-S: 1
Y-S: 1

50:Y=SIN X*ABS X



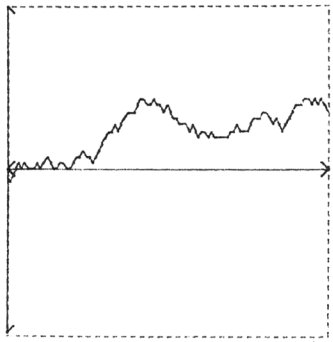
RECTANGULAR COORDINATES
X: -30, 30
Y: -30, 30
INC: 0.8
X-S: 5
Y-S: 5

50:Y=TAN X



RECTANGULAR COORDINATES
X: -6.283, 6.283
Y: -50, 50
INC: 0.016755
X-S: 0.31
Y-S: 10

50:Y=RND 3-2+LY:L Y=Y



RECTANGULAR COORDINATES
X: 0, 100
Y: -25, 25
INC: 1

FROM THE WATCH POCKET

Well, they (Sharp) finally broke down and released information on machine language programming of the PC-1500 (which should also apply to the Radio Shack PC-2). *PCN* will take the credit (thank you) for virtually embarrassing them into acknowledging such capability. Remember, we (thanks particularly to *Norlin Rober*) published the machine codes for almost the entire instruction set about a year ago and have provided a good deal of supplemental material on various aspects of the machine over the past year.

In any event, the new *Sharp Pocket Computer PC-1500 Technical Reference Manual* verifies a lot of what *PCN* has published during the past year. And, it adds a lot more information that will be of interest to PC-1500/PC-2 afficionados.

The book starts off with an overview of the LH5801 CPU, including a block diagram, a summary of its internal registers and status flags and the CPU chip pin-out. There is a detailed description of each signal line. Information is provided on the internal timer, the interrupt system, and special functions (such as reset and wait). CPU execution sequences and timing diagrams are also provided. The instruction set is discussed in detail. A diagram illustrating how CPU registers are affected by each type of instruction is provided in this section. Sharp provides their own set of mnemonics to describe the instruction set.

Note: Be advised that PCN *will continue to use* Rober Mnemonics *in its articles and machine language programs.*

There is a substantial chapter that describes and discusses the application of the LH5810/5811 I/O port controller. Much of this information is new material (even for *PCN* subscribers).

There is a chapter devoted to describing the overall PC-1500 system. This includes information on power supplies (such as in the actual PC-1500, the CE-150 printer/cassette interface, the CE-158 RS-232 interface and the CE-159 RAM/ROM memory module). There is also information on the connectors used in the system (signal names on a pin-out basis), comprehensive system memory maps, and keyboard matrix and key code charts.

Another chapter discusses BASIC commands that were not mentioned in the original PC-1500 operating manual. *PCN* subscribers are already aware of most of the information provided in this section relating to such capabilities as PEEK, POKE, STATUS, CALL, CSAVE M and CLOAD M. Later in this section there is information on how variables are stored in memory and some system subroutine calls. A number of these machine language calls are already known to *PCN* subscribers, but there is some new information of interest to machine language programmers here, too.

The POCKET COMPUTER NEWSLETTER is Available!

By Subscription Only: for a calendar year period (January – December). You get all issues published to date for the calendar year in which you subscribe, at the time you subscribe.

MC/VISA Phone Subscriptions: (203) 888-1946

☐ 1982 Regular Subscriber (Issues 11 – 20). $30.00 in U.S. (U.S. $36.00 to Canada. U.S. $45.00 elsewhere.)

☐ 1982/83 Subscriber (Issues 11 – 30). $60.00 in U.S. (U.S. $72.00 to Canada. U.S. $90.00 elsewhere.)

☐ 1983 Regular Subscriber (Issues 21 – 30). $36.00 in U.S. (U.S. $42.00 to Canada. U.S. $50.00 elsewhere.)

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER.

Thank you for your remittance.

Name: _____

Addr: _____

City: _____ State: ___ Zip: _____

MC/VISA #: _____ Expires: _____

Signature: _____

P.O. Box 232, Seymour, CT 06483

Another chapter provides some machine language programming examples. There is a binary to hexadecimal conversion routine, a display inversion routine, a routine that will shift the display right or left by one dot and a few other relatively simple procedures.

The appendices includes schematic diagrams of the PC-1500, the CE-150, CE-151, CE-153, CE-155, CE-158 and CE-159.

While this book has a lot of information (about 160 pages in large 8-1/2 by 11 inch format), it is not particularly easy to digest. Much of the English appears as though someone was trying to translate directly from Japanese. You will have to re-read many passages in order for the material to make sense. *This is not a tutorial on machine language programming!* It is assumed that you already know how to perform such programming. However, if your interest lies in the area of performing such programming on a PC-1500/PC-2, then this book is practically a must as a reference manual. Ditto if you want to do interfacing to equipment of your own design, etc.

Where can you get a copy? My understanding is that they are only being made available through authorized Sharp distributors. Mort Rosenstein at **Atlantic NorthEast Marketing**, *P.O. Box 921, Marblehead, MA 01945*, (phone (617) 639-0285) says he has copies available. The price from him is $20.00 for delivery in the United States. Contact him for pricing information on deliveries outside the U.S.

First ROM Program Library Available

The first of the long-awaited ROM software libraries distributed by Sharp are now available in the United States. I have actually seen a CE-502A General Statistics Module in operation. My first impression was that it looked pretty good. We plan on having a thorough review of the package in the next issue of *PCN*. Check with your local Sharp distributor on pricing. I understand the list price of the CE-502A is $69.95, but you may be able to find it for $10 or $15 less.

Incidently, the rest in the series of modules are due to be released this summer. The Finance package is due at any moment. A Circuit Analysis, Business Graphics and a second Statistics module are due in June. July should yield general Graphics, Electrical Engineering and Mathematics modules. Keep checking with your local distributor.

Here Come More Portables

Sharp has announced that it will begin delivering a portable computer named the PC-5000 this fall. The battery operated unit, about the size of a small portable typewriter, will sport an unusual liquid-crystal display. It will be capable of presenting 8 lines of 80 characters in the text mode or 640 by 80 dots in a bit-mapped graphics mode.

There are other unique features. It will be equipped with 128K of RAM expandable to 256K! It also has 128K (bytes) of bubble memory and can be equipped with up to 128K of ROM.

The machine will use an 8088 microprocessor. It can be fitted with a high-density dot-matrix thermal/impact printer. This unusual printer can operate in a strictly thermal mode using thermal paper. Alternately, when a carbon ribbon is inserted, it can print in an impact mode on plain bond paper! The unit can also be equipped with a modem/auto-dialer.

Optional double-sided, double-density disk drives, capable of storing up to 320 kilobytes (each) will also be made available for the unit.

The PC-5000 will apparently be equipped with Microsoft Basic and a bundle of software from Sorcim, including a spreadsheet package, word processing, data base management and a communications program.

The PC-5000 will probably be priced at about $2000.00 initially in the United States. The optional printer will probably go for about $400. RAM expansion modules are reported to be in the vicinity of $150.00 for 64K.

I will have more information on the PC-5000 in the next issue.

Casio also plans to introduce a notebook-sized portable computer near the end of this summer. I bet there will be lots of others!

New Sinclair ZX-81/Timex 1000 Software

If you are looking for ZX-81/Timex 1000 software, write to **American Micro Products**, *705 North Bowser, Richardson, TX 75081*. They have just announced a new Timex/Sinclair line. — *Nat Wadsworth, Editor*